Appendices

Appendix A – Clinical Assessment of Ultrasound Images

Clinical assessment

Digital watermarking introduces identifiers to guard against false ownership claims and fabrication. Through visual inspection by the naked eye, these images appear to be unaltered and therefore medical diagnoses should not be any different regardless of the presence or absence of the identifiers. Technically speaking, the image pixels are preserved despite the introduction of the identifiers and therefore the assumption that the clinical diagnoses remain unchanged is technically sound. However, clinical assessment of these images would in some way add further evidence to the conclusions reached so far, but more importantly, such a study would reduce anxieties and fears that may arise among clinicians as the result of this technique.

Methodology and Statistical Analysis

This study involves subjecting assessors to two sets of images, the original (group O) and those digitally watermarked (group DW). Both groups view ultrasound images that essentially are similar, except the latter has been digitally watermarked with this new technique. Group 0 would be regarded as a control against for group DW to be compared against. This study is conducted in a blind manner in that the assessors do not know which of the images to be assessed has been watermarked. The assessors will consist of four radiologists at consultant level to achieve authority and consistency in assessment, who would therefore be familiar with ultrasound images used in clinical practise.

Fifteen images are used as controls in group O. The same images are digitally watermarked to represent group DW. All images (O and DW) would be randomly

assigned to the assessors who are blind to the group the image belongs to. Each image will carry a clinical stem to prompt clinical diagnosis. In cases where a clinical diagnosis is not possible, reasons for this are sought, but which could be owing to poor image quality. The assessors are also given the opportunity to add comments on all aspects of the image being assessed should they need to do so.

All images assessed would be re grouped into groups O and DW and a Chi Square test is employed to detect any significant difference between them. A value $P<0.05$ is taken as the level of significance. Further analysis would also be carried out on images (if incorrectly diagnosed) to explain this finding.

The study would also look into comments made by assessors in all aspects of the images, as these comments would also form the basis of any conclusion formed from this clinical assessment.

List of images for evaluation:

1. Abdominal aortic aneurysm
2. Adrenal mass
3. Liver cysts
4. Liver metastases
5. Cholecystitis / cholelithiasis
6. Achilles tendon tear
7. Forearm abscess
8. Muscle mass
9. Patellar tendon tear
10. Rotator cuff tear
11. Breast cyst
12. Abnormal endometrium
13. Adnexal mass
14. Ovarian cyst
15. Greater saphenous vein thrombosis.

CLINICAL ASSESMENT OF ULTRASOUND IMAGES

Investigator: Jasni M Zain
Brunel University

Thank you for agreeing to participate in this study.

The aim is to find out whether or not embedded ultrasound images alter clinical diagnosis when compared to the original ones.

In this study, ultrasound images underwent a process called watermarking, mainly to add security during image transfer. By visual inspection, the images do not change as the result of the process; this study will go one step further by subjecting these images to objective clinical assessment.

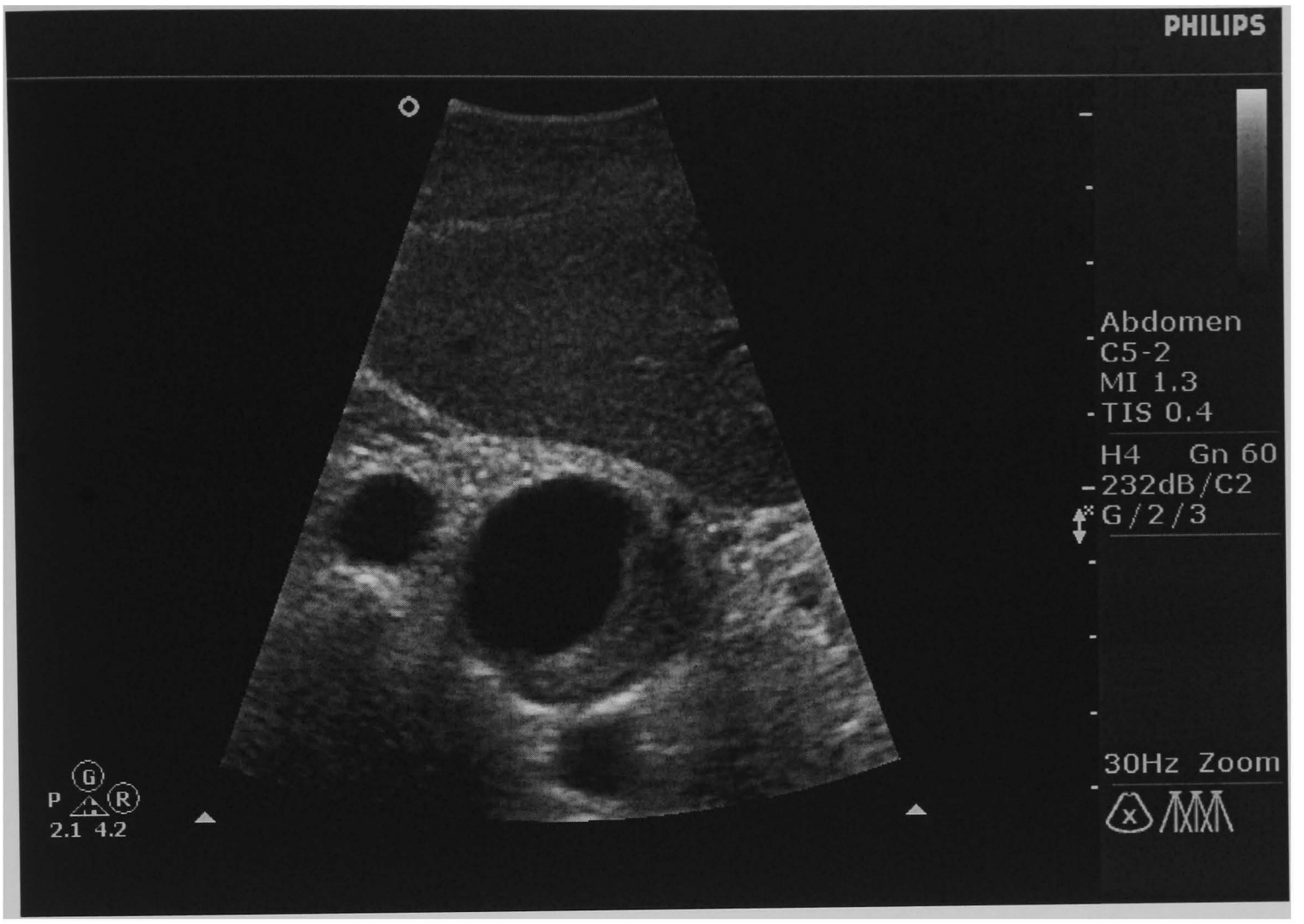
You will be given 10 ultrasound images, a mixture of the original and the embedded, each with a brief clinical summary to help you arrive at the most likely diagnosis. You will not be able to differentiate whether or not these images have been embedded.

Your task is to enter the most likely diagnosis based on the clinical summary and the ultrasound image provided. Please do not write a descriptive report. If you cannot arrive at a single diagnosis, please choose a box to state the reason.

Thank you for your kind help.

Clinical summaries and the ultrasound images

1. Abdominal ultrasound of a 72-year-old man with chronic abdominal discomfort.



The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

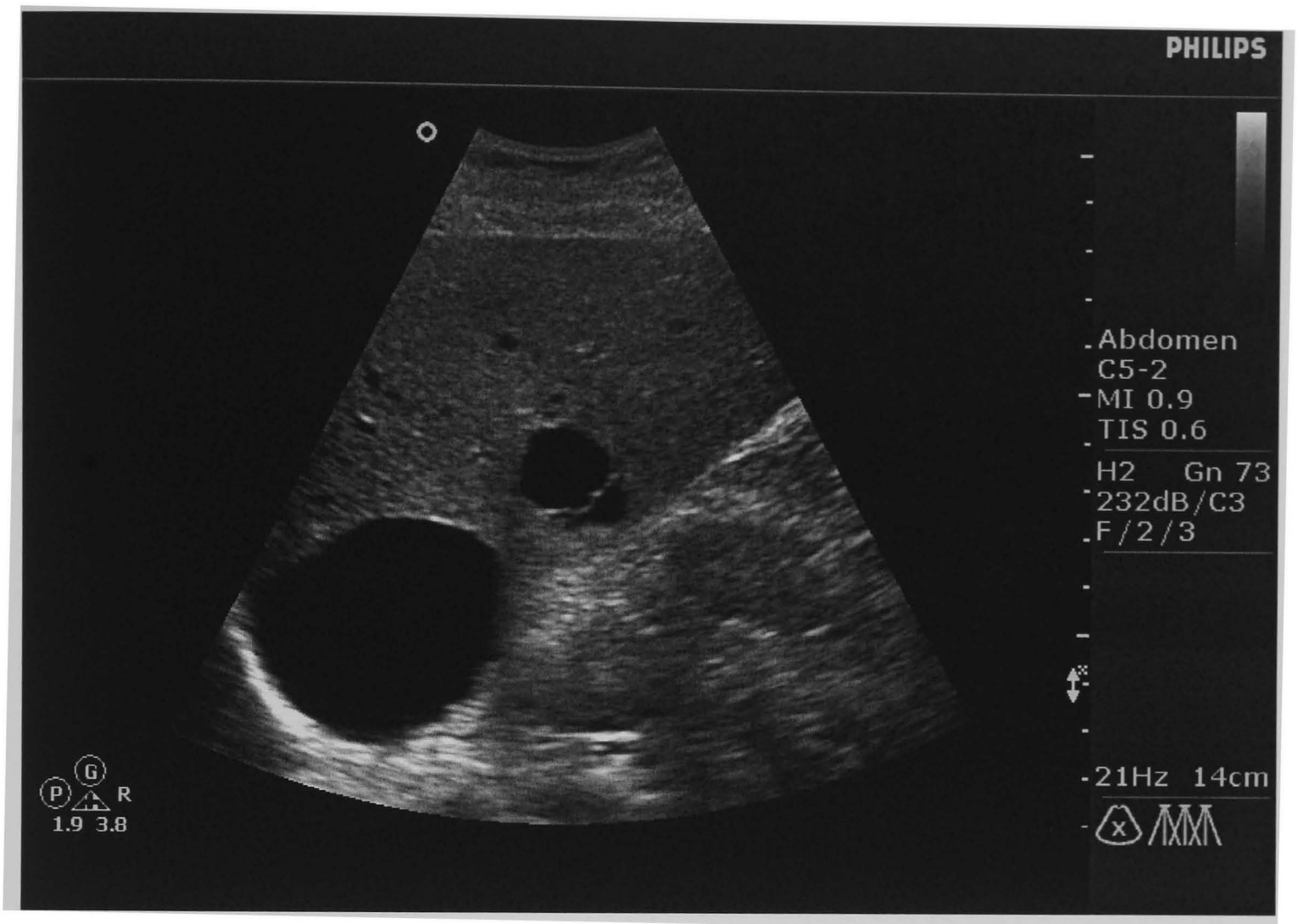
2. A screening abdominal ultrasound of a woman with confirmed lung malignancy showing the left adrenal.

The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

3. A young woman with one-day history of right upper quadrant pain that resolved on arrival to the casualty department. The liver functions tests and total white cell count were all normal. This is an ultrasound image of the liver.



The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

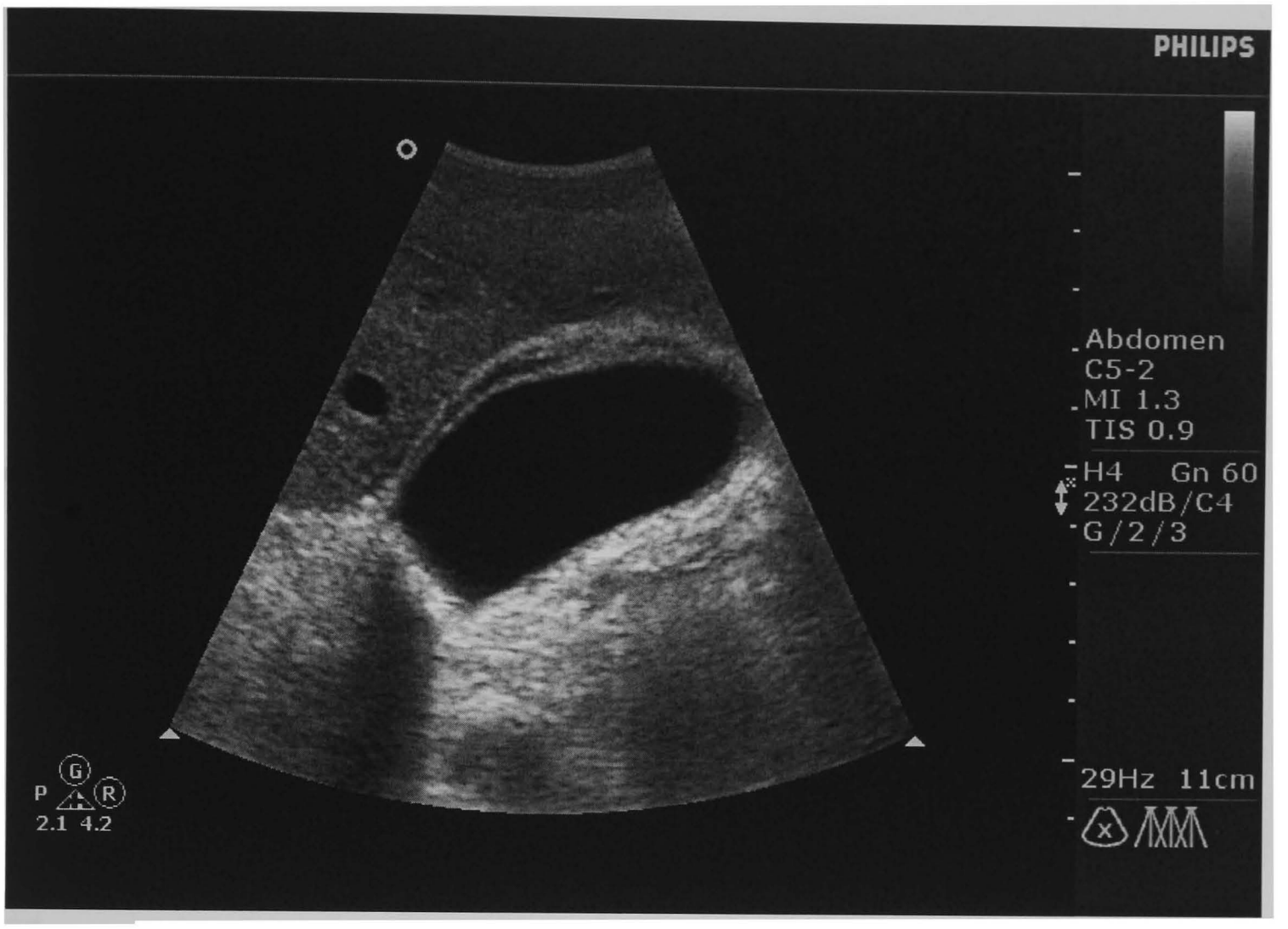
4. A liver ultrasound of a man with adenocarcinoma of her left lung.

The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

5. An obese female patient with 2 days history of fever and constant right upper quadrant pain.

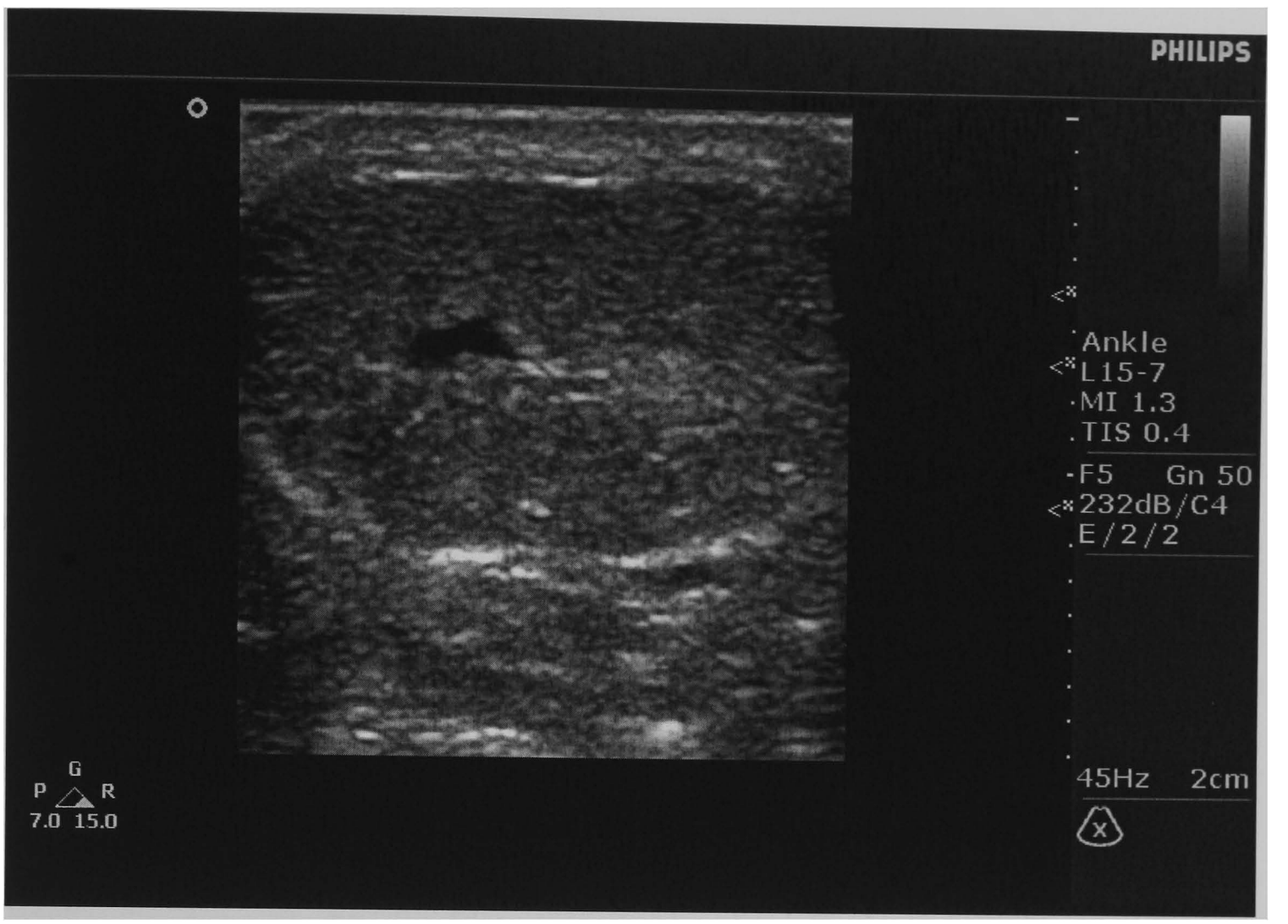


The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

6. A left heel ultrasound of an amateur rugby player who could not walk following a tackle.

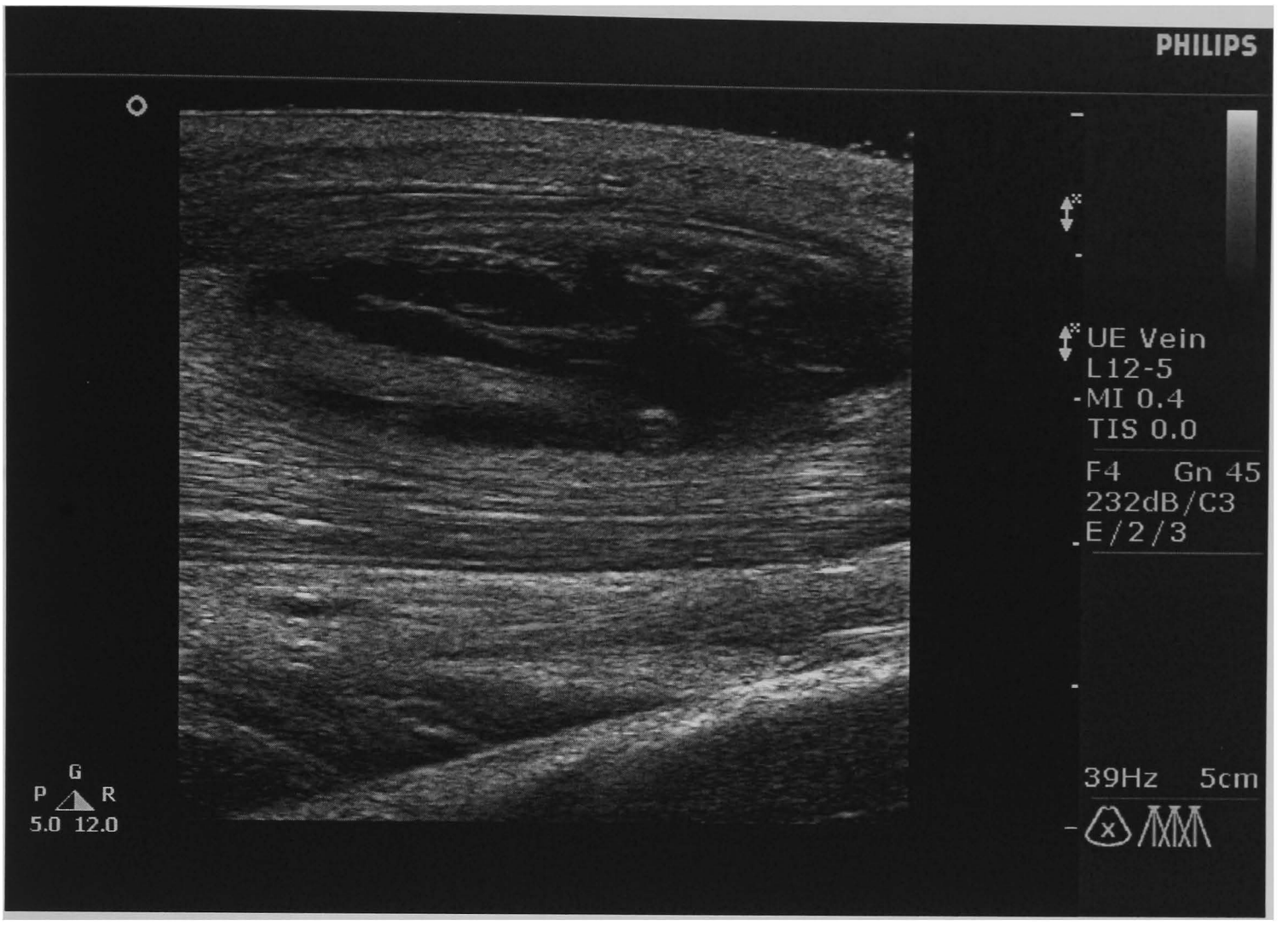


The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

7. A poorly controlled diabetic man with a swollen and tender forearm. Blood tests showed elevated white cells with neutrophils predominance. This is the ultrasound of his forearm.



The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

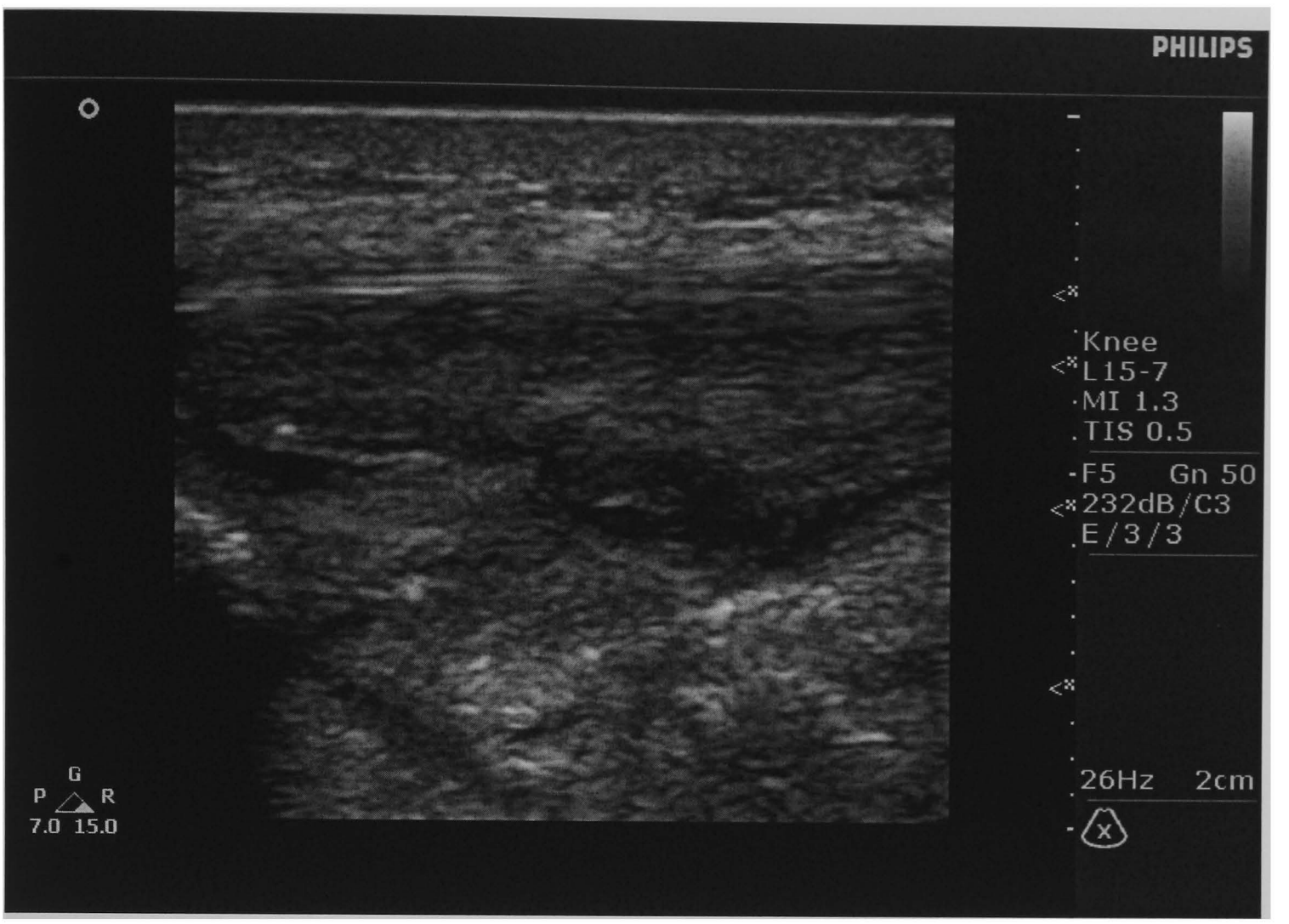
8. A middle-aged man with recent history of thigh swelling. This image was from the swollen area.

The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

9. A knee ultrasound of a man with a tender kneecap following a football game.

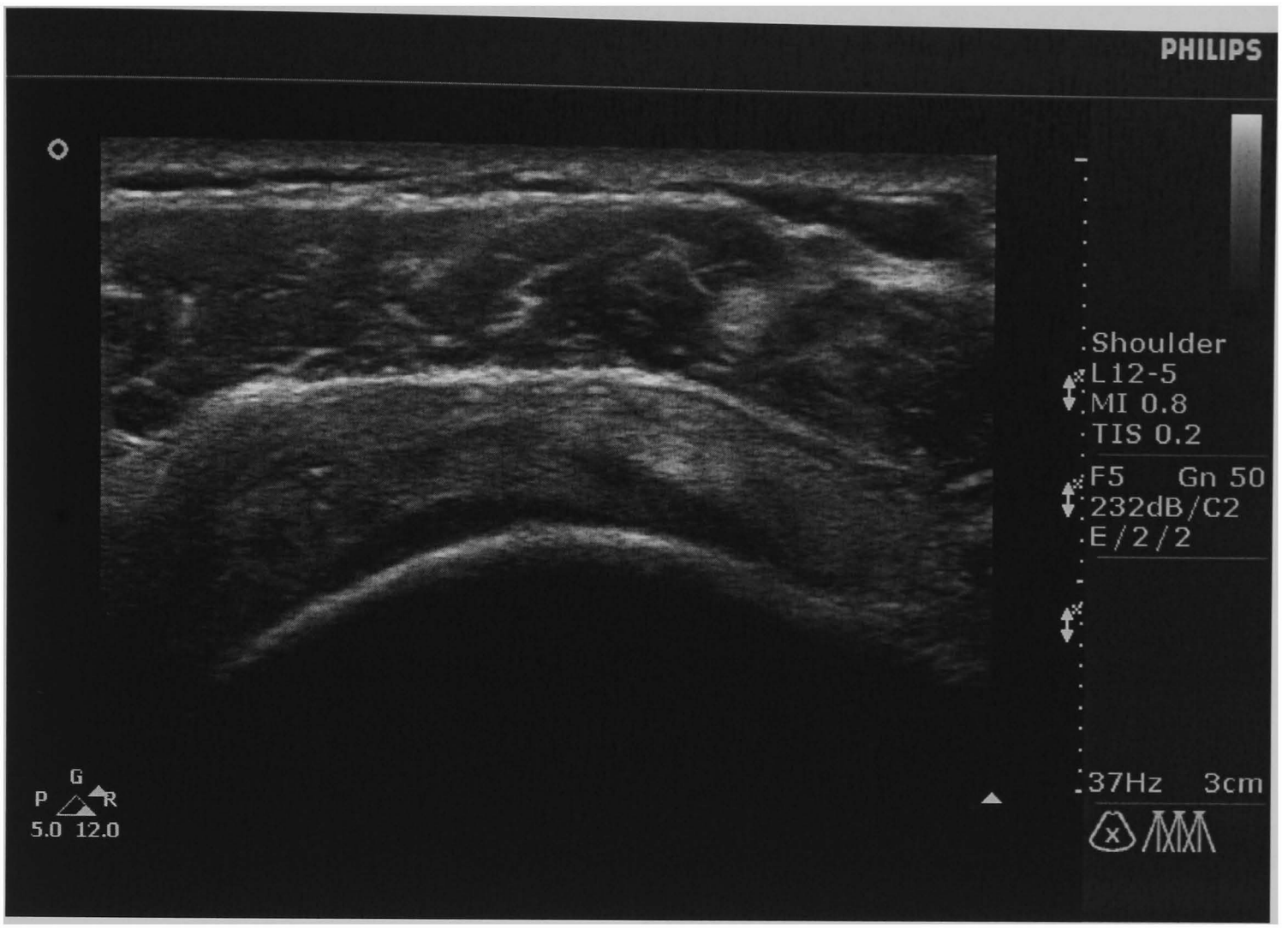


The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

10. A shoulder ultrasound done on a javelin thrower with severe shoulder pain following a big javelin throw.



The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

11. A breast ultrasound of an asymptomatic woman.

The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

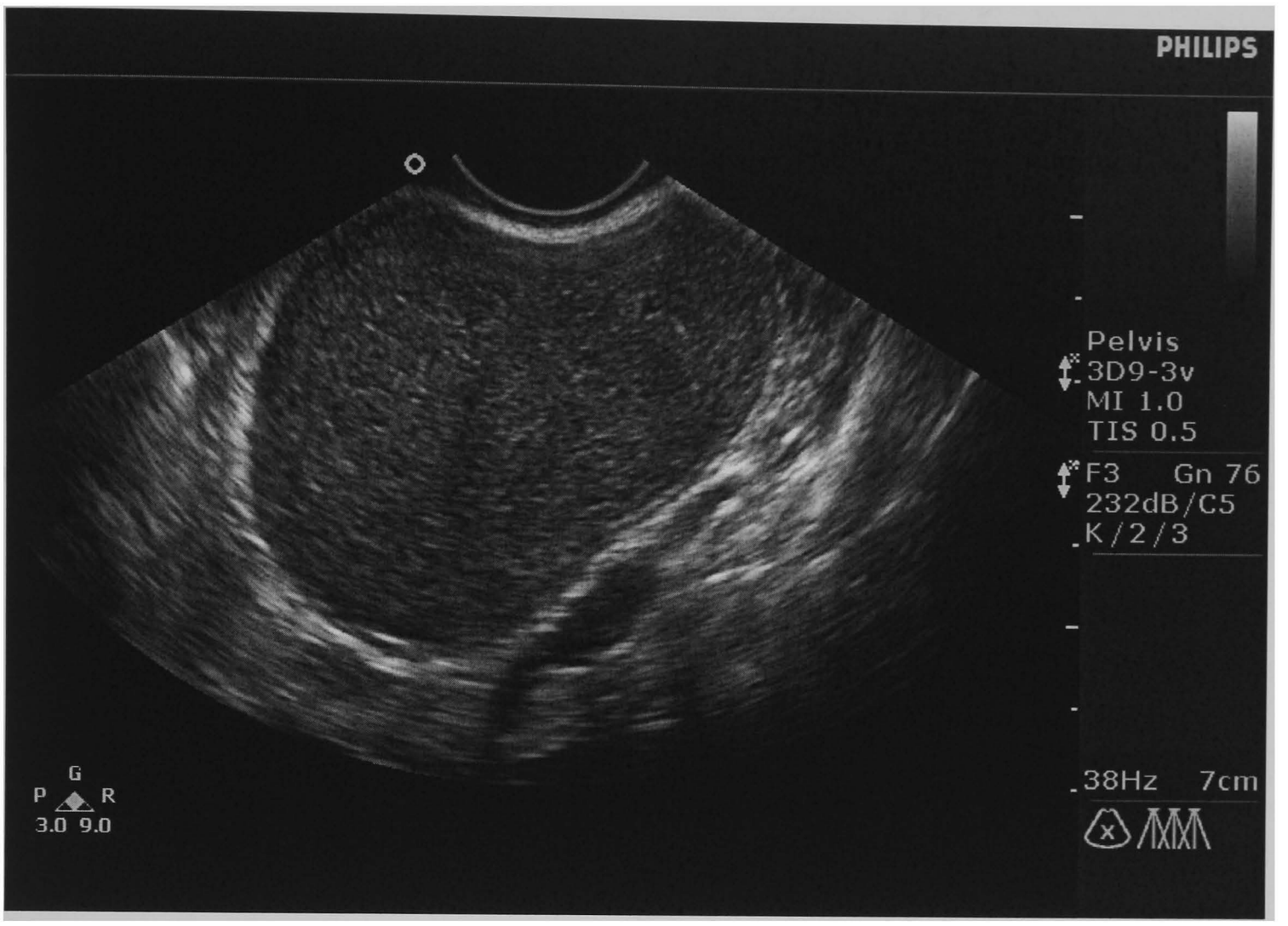
12. A lady with frequent heavy periods underwent a pelvic ultrasound.

The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

13. An adnexal image from a pelvic ultrasound of an elderly lady with malignant ascites.



The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

14. A young woman with an intermittent left iliac fossa pain. This image is from her left ovary.

The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

15. A right inner thigh ultrasound of an obese woman with chronic varicose veins and previous history of right deep venous thrombosis. She presented with an acutely swollen right leg.

The most likely diagnosis is

Please choose the appropriate box if no diagnosis is entered

Don't know	
Inadequate clinical information	
Poor image quality	
Two or more diagnoses are equally likely	
Others	

Appendix B – Program Listing

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Function:     Calculates the PSNR (Peak Signal to Noise Ratio)
%                 of images A and A', both of size MxN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function A = psnr(image,image_prime,M,N)

   % convert to doubles
   image=double([image]);
   image_prime=double([image_prime]);

   % avoid divide by zero nastiness
   if (sum(sum(image-image_prime)) == 0)
      error('Input vectors must not be identical')
   else
      psnr_num=M*N*max(max(image.^2));              % calculate numerator
      psnr_den=sum(sum(image-image_prime).^2);      % calculate denominator
      A=psnr_num/psnr_den;                          % calculate PSNR
   end

return
```

```
 function [PSNR,mse]=psnr(X,Y)
% function [PSNR,mse]=psnr(X,Y)
% Peak signal to noise ratio of the difference between images and the
%mean square error
%    If the second input Y is missing then the PSNR and MSE of X itself
% becomes the output (as if Y=0).

if nargin<2, D=X;
else
  if any(size(X)~=size(Y)), error('The input size is not equal to each other!'); end
  D=X-Y;
end

mse=sum(D(:).*D(:))/prod(size(X))
PSNR=10*log10(255^2/mse)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name:          Jasni Zain
% Project:      Strict Authentication Watermarking
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Hare = dicomread ('JZI944SA.DCM');
info= dicominfo ('JZI944SA.DCM');
%F15 = imread('nhs.jpg', 'jpeg');
%n =4; % Number of bits to replace 1 <= n <= 7
%Hare= rgb2gray(Hare);
Hare = double(Hare);
%a=mat2str(Hare);
%z= md5(a)
Hare1 = Hare;
for i = 241:248
   for j =9:16
      for b=2:-1:1

         Hare1(i,j)= bitset(Hare1(i,j),b,1);

      end
   end
end
for i = 241:248
   for j =17:24
      for b=2:-1:1

         Hare1(i,j)= bitset(Hare1(i,j),b,1);

      end
   end
end
for i = 241:248
   for j =25:32
      for b=2:-1:1

         Hare1(i,j)= bitset(Hare1(i,j),b,0);
         %k=k+1;
      end
   end
end
for i = 241:248
   for j =33:40
      for b=2:-1:1

         Hare1(i,j)= bitset(Hare1(i,j),b,0);
         %k=k+1;
```

```
      end
   end
end

a=Hare1(241:248,9:40)
%Stego = uint8(double(RemoveLSB(Hare, n)) + double(F15) / 2^(8 - n));

%Extracted = uint8(double(RemoveMSB(Stego, n))*2^(8-n));
psnr_num=800*600*max(max(Hare.^2));              % calculate numerator
      psnr_den=sum(sum(Hare-Hare1).^2);      % calculate denominator
      psnr=psnr_num/psnr_den
Hare1=uint8(Hare1);
dicomwrite(Hare1, 'c:\temp\wm5.dcm');
imview(Hare1, [])
%figure,imshow(Extracted)
Hare2=dicomread('c:\temp\wm5.dcm');

Hare2 = double(Hare2);
Hare3 = Hare2;
Arr1=[];
for i = 241:248
   for j =9:40
      for b=2%2:-1:1

         Arr2= [bitget(Hare3(i,j),b)];
         Hare3(i,j)=bitset(Hare3(i,j),b,0);
         Arr1=[Arr1 Arr2];
      end
   end
end
Arr1
Hare3=uint8(Hare3);
%x= mat2str(Hare3);
%y= md5(x)
%dicomwrite(Hare3,'c:\temp\result4.dcm');
%imview(Hare3, [])
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  This program implements the Secure Hash Standard SHA-256 as set forth by
%  the Federal Information Processing Standards Publication 180-2.
%
%
%
%Inputs:
%         1. Input File - name of the input file. The file must be text
%           file in ASCII format.
%         2. Output File - name/location of the output file.
%
%Outputs:
%         1. Output File- The final hash value is
%           placed in this file as a Hex value on the first line.
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sha256()

%Ask for user input
fname=input('Input File (in ASCII format)? ','s');
hash_foutname=input('Output File for SHA256 Hash? ','s');

%Open the input file and get the first line of data
fid=fopen(fname);
M = fread(fid);
fclose(fid);

%Convert the input message from ASCII to 8-bit binary values for each character
% M=dec2bin((abs(input(1))),8);
% for i = 2:length(input)
%     M=strcat(M,dec2bin(abs(input(i)),8));
% end

%Get Constants - K256 and initial hash values, H0
[K256 H]=constants(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                PREPROCESSING SECTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%PAD THE MESSAGE

%Calculate the number of zeros needed to pad the message up to 448
len=8*length(M);
k=mod(448-mod(len,512)-1,512);

if (k > 0)
   M(length(M)+1)=128;
end

for i=2:(k+1)/8
   M(length(M)+1)=0;
end

%Append the bit value of the length of the message to fill up to 512
len_bin=dec2base(len,2,64);
for i=1:8
   M(length(M)+1) = bin2dec(len_bin(1,(8*i-7):(8*i)));
end


%PARSING THE PADDED MESSAGE

%Calculate the number of blocks in the message
N=length(M)/64;

%Split the message into N 512-bit blocks of message
%Each N block has 16 32-bit blocks
cnt = 1;
for i=1:N
   for j=1:16

M_parsed(i,j)=bitshift(M(cnt),24)+bitshift(M(cnt+1),16)+bitshift(M(cnt+2),8)+M(cnt+
3);
      cnt = cnt + 4;
   end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                PROCESSING SECTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Process each 512 bit block of Message individually
for i = 1:N
```

```
%PREPARE THE MESSAGE SCHEDULE
%The first 16 blocks of message schedules are 32-bit blocks of the N block
for t = 1:16
   W(t)=M_parsed(i,t);
end

%The next 48 message schedules are calculated as follows from the initial 16
message schedules
for t = 17:64

   W(t) = add4num32(gam1(W(t-2)),W(t-7),gam0(W(t-15)),W(t-16));

end

%Intialize the eight working variables to initial hash values
a = H(i,1);
b = H(i,2);
c = H(i,3);
d = H(i,4);
e = H(i,5);
f = H(i,6);
g = H(i,7);
h = H(i,8);

%Compute all 64 iterations of the eight working variables
for t = 1:64

   T1 = add5num32(h,eps1(e),Ch(e,f,g),K256(t),W(t));
   T2 = mod(eps0(a) + Maj(a,b,c),2^32);
   h = g;
   g = f;
   f = e;
   e = mod((d + T1),2^32);
   d = c;
   c = b;
   b = a;
   a = mod((T1 + T2),2^32);

end

%Compute the i-th hash values for N block
H(i+1,1) = mod((a + H(i,1)),2^32);
H(i+1,2) = mod((b + H(i,2)),2^32);
H(i+1,3) = mod((c + H(i,3)),2^32);
H(i+1,4) = mod((d + H(i,4)),2^32);
H(i+1,5) = mod((e + H(i,5)),2^32);
```

```
    H(i+1,6) = mod((f + H(i,6)),2^32);
    H(i+1,7) = mod((g + H(i,7)),2^32);
    H(i+1,8) = mod((h + H(i,8)),2^32);

end

%Open the output file and store hex values of each hash as one line

fid=fopen(hash_foutname,'at');
fprintf(fid,'%s',dec2hex(H(N+1,1),8));
fprintf(fid,'%s',dec2hex(H(N+1,2),8));
fprintf(fid,'%s',dec2hex(H(N+1,3),8));
fprintf(fid,'%s',dec2hex(H(N+1,4),8));
fprintf(fid,'%s',dec2hex(H(N+1,5),8));
fprintf(fid,'%s',dec2hex(H(N+1,6),8));
fprintf(fid,'%s',dec2hex(H(N+1,7),8));
fprintf(fid,'%s',dec2hex(H(N+1,8),8));
fclose(fid);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                FUNCTIONS SECTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Function: gam0
%
% Defined SHA256 function
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = gam0(x)

result1=rotr(x,7);
result2=rotr(x,18);
result3=shr(x,3);
result = bitxor(bitxor(result1,result2),result3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
```

```
%
%  Function: gam1
%
%  Defined SHA256 function
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = gam1(x)

result1=rotr(x,17);
result2=rotr(x,19);
result3=shr(x,10);
result = bitxor(bitxor(result1,result2),result3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%  Function: eps0
%
%  Defined SHA256 function
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = eps0(x)

 result1=rotr(x,2);
 result2=rotr(x,13);
 result3=rotr(x,22);
 result = bitxor(bitxor(result1,result2),result3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%  Function: eps1
%
%  Defined SHA256 function
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = eps1(x)

result1=rotr(x,6);
result2=rotr(x,11);
result3=rotr(x,25);
result = bitxor(bitxor(result1,result2),result3);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Function: shr
%
% Shifts a binary number x positions to
% the right, placing '0' values in the
% x left positions
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = shr(x,n)

result = bitshift(x,-n,32);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Function: rotr
%
% Shifts a binary number x positions to
% the right, rotating the shifted values
% back into the left.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = rotr(x,n)

result = bitor(bitshift(x,-n,32),(bitshift(x,32-n,32)));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% function: add4num32
%
% Adds 4 32-bit numbers
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = add4num32(x1,x2,x3,x4)

result=mod(x1+x2+x3+x4,2^32);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%  function: add5num32
%
%  Adds 5 32-bit numbers
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = add5num32(x1,x2,x3,x4,x5)

result=mod(x1+x2+x3+x4+x5,2^32);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%  function: Ch
%
%  defined SHA256 function
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = Ch(x,y,z)

 temp1 = bitand(x,y);
 temp2 = bitand(bitcmp(x,32),z);
 result = bitxor(temp1,temp2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%  function: Maj
%
%  defined SHA256 function
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [result] = Maj(x,y,z)

temp1 = bitand(x,y);
temp2 = bitand(x,z);
temp3 = bitand(z,y);
result = bitxor(bitxor(temp1,temp2),temp3);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%  function: constants
%
%  produces SHA256 constants
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [K256,H] = constants(temp);

%SHA-256 Constant Definitions
%1
K256(1)=1116352408;K256(2)=1899447441;K256(3)=3049323471;K256(4)=3921009
573;
K256(5)=961987163;K256(6)=1508970993;K256(7)=2453635748;K256(8)=28707632
21;
%2
K256(9)=3624381080;K256(10)=310598401;K256(11)=607225278;K256(12)=142688
1987;
K256(13)=1925078388;K256(14)=2162078206;K256(15)=2614888103;K256(16)=324
8222580;
%3
K256(17)=3835390401;K256(18)=4022224774;K256(19)=264347078;K256(20)=6048
07628;
K256(21)=770255983;K256(22)=1249150122;K256(23)=1555081692;K256(24)=1996
064986;
%4
K256(25)=2554220882;K256(26)=2821834349;K256(27)=2952996808;K256(28)=321
0313671;
K256(29)=3336571891;K256(30)=3584528711;K256(31)=113926993;K256(32)=3382
41895;
%5
K256(33)=666307205;K256(34)=773529912;K256(35)=1294757372;K256(36)=13961
82291;
K256(37)=1695183700;K256(38)=1986661051;K256(39)=2177026350;K256(40)=245
6956037;
%6
K256(41)=2730485921;K256(42)=2820302411;K256(43)=3259730800;K256(44)=334
5764771;
K256(45)=3516065817;K256(46)=3600352804;K256(47)=4094571909;K256(48)=275
423344;
%7
K256(49)=430227734;K256(50)=506948616;K256(51)=659060556;K256(52)=883997
877;
K256(53)=958139571;K256(54)=1322822218;K256(55)=1537002063;K256(56)=1747
873779;
```

```
%8
K256(57)=1955562222;K256(58)=2024104815;K256(59)=2227730452;K256(60)=236
1852424;
K256(61)=2428436474;K256(62)=2756734187;K256(63)=3204031479;K256(64)=332
9325298;

%Intial Hash Values

H(1,1)=1779033703;
H(1,2)=3144134277;
H(1,3)=1013904242;
H(1,4)=2773480762;
H(1,5)=1359893119;
H(1,6)=2600822924;
H(1,7)=528734635;
H(1,8)=1541459225;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name:          Jasni Zain
%Project:       Image relocation using toral automorphism
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

% save start time
start_time=cputime;

blocksize=4;            % set the blocksize

% read in the cover object
file_name='sig4.bmp';
cover_object=(imread(file_name));
%cover_object=rgb2gray(cover_object);
cover_object=double(cover_object);
% determine size of cover image
Mc=size(cover_object,1);          %Height
Nc=size(cover_object,2);          %Width
%ABB=[];
Br=floor(Nc/blocksize);  % Blocks per row
Bc= floor(Mc/blocksize); % Blocks per column

numblock= Br*Bc; % number of blocks
k=max(primes(numblock/2));


ABB=[];
for A= 1:numblock
   AB = mod((k*A), numblock)+1; %mapping the blocks
   ABB=[ABB AB];
end
B=[1:numblock];
mapA= [B;ABB];

mapB=[];mapBB=[];
for i= 1:numblock
   mapB(1,mapA(2,i))=mapA(2,i); %mapping the blocks
   mapB(2,mapA(2,i))=mapA(1,i);

end
new_image= cover_object;
x=1; y=1;
for i=1:numblock
    % numbering block
```

```
   block=cover_object(y:y+blocksize-1,x:x+blocksize-1);

   targetblock= mapA(2,i);
   rownum=round( ceil(targetblock/Br));
   colnum= round(targetblock-(rownum-1)*Br);
   xs=(rownum-1)*blocksize+1;
   ys=(colnum-1)*blocksize+1;


   if (x+blocksize) > Nc
    if y+blocksize < Mc
       x=1;
       y=y+blocksize;
    end
     else
       x=x+blocksize;

    end
       new_image(xs:xs+blocksize-1,ys:ys+blocksize-1)=block;
  end

%sub-block watermark generation
toral_image=uint8(new_image);
imwrite(toral_image,'toraltest3.bmp','bmp');

% display processing time
%elapsed_time=cputime-start_time,

% display psnr of watermarked image
%psnr=psnr(cover_object,watermarked_image),

% display watermarked image
figure
imshow(toral_image)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name:             Jasni Zain
%Project:     Spread Embedding for Tamper detection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

% save start time
start_time=cputime;

blocksize=8;              % set the blocksize
percent= 50;            %set tamper percentage

% read in the cover object
file_name='ustest.bmp';
cover_object=(imread(file_name));
%cover_object=rgb2gray(cover_object);
cover_object=double(cover_object);
% determine size of cover image
Mc=size(cover_object,1);            %Height
Nc=size(cover_object,2);            %Width
areatam= Mc*Nc*percent;             % area of tamper
%ABB=[];
Br=floor(Nc/blocksize);  % Blocks per row
Bc= floor(Mc/blocksize); % Blocks per column

numblock= Br*Bc; % number of blocks
numtamblk= floor(numblock*percent/100);

factam=floor(100/percent);  %factor of tampered block
k=max(primes(numblock/2));
tampered_image=cover_object;
for i= 1:numtamblk
   targetblock= i;
   rownum=round( ceil(targetblock/Br));
   colnum= round(targetblock-(rownum-1)*Br);
   ys=(rownum-1)*blocksize+1;
   xs=(colnum-1)*blocksize+1;
   startblock=cover_object(ys:ys+blocksize-1,xs:xs+blocksize-1);%start of target block
      for ii=1:blocksize
         for jj= 1:blocksize
            startblock(ii,jj)=255;
         end
      end
      tampered_image(ys:ys+blocksize-1,xs:xs+blocksize-1)= startblock;
end
tampered_image=uint8(tampered_image);
```

```
imwrite(tampered_image,'tamper250.bmp','bmp');


% display watermarked image
figure
imshow(tampered_image)
title('Tamper detect')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name:                  Jasni Zain
%Project:       Embedding for Tamper detection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

% save start time
start_time=cputime;

blocksize=8;                % set the blocksize


% read in the cover object
file_name='ultrasound2.jpg';
cover_object=(imread(file_name));
cover_object=rgb2gray(cover_object);
cover_object=double(cover_object);
% determine size of cover image
Mc=size(cover_object,1);            %Height
Nc=size(cover_object,2);            %Width

Br=floor(Nc/blocksize);  % Blocks per row
Bc= floor(Mc/blocksize); % Blocks per column
ABB=[];
numblock= Br*Bc; % number of blocks
k=max(primes(numblock/2));

for A= 1:numblock
   AB = mod((k*A), numblock)+1; %mapping the blocks
   ABB=[ABB AB];
end
B=[1:numblock];
mapA= [B;ABB];
mapB=[];
for i= 1:numblock
   mapB(1,mapA(2,i))=mapA(2,i); %mapping the blocks
   mapB(2,mapA(2,i))=mapA(1,i);

end
mapB;

% generate shell of watermarked image
watermarked_image=cover_object;
x=1;
y=1;
```

```
% process the image in blocks

for i = 1:numblock

   % numbering block
   block=cover_object(y:y+blocksize-1,x:x+blocksize-1);
   cover= RemoveLSB(block,1); % reset the LSB to 0
   AvgB=round(sum(sum(cover))/(blocksize*blocksize)); % average of block
   targetblock= mapB(2,i);
   rownum=round( ceil(targetblock/Br));
   colnum= round(targetblock-(rownum-1)*Br);
   ys=(rownum-1)*blocksize+1;
   xs=(colnum-1)*blocksize+1;
   startblock=cover_object(ys:ys+blocksize-1,xs:xs+blocksize-1);%start of target block
   cover1=RemoveLSB(startblock,1);

   % prepare sub-block
   x1=1;
   y1=1;
   for sub= 1:4
   subblok= cover(y1:y1+(blocksize/2)-1, x1:x1+(blocksize/2)-1);
   AvgBs=round(sum(sum(subblok))/(blocksize/2)*(blocksize/2));
   if AvgBs >= AvgB
      v=1;
   else v=0;
   end
   %parity
   par=0;
   embedbit=[];

    for b=8:-1:2
      bit=bitget(AvgBs,b);
         if bit==1
         par=par+1;
         end
      end

   if rem (par,2)==0 %even
      p= 1;
   else p=0;
   end
   substart=cover1(y1:y1+(blocksize/2)-1, x1:x1+(blocksize/2)-1);
   AvgBc=round((sum(sum(substart)))/((blocksize/2)*(blocksize/2)));
      for b=8:-1:2
      bit=bitget(AvgBc,b);
      embedbit=[embedbit bit];
      end
```

```
   embedbit=[v p embedbit]; %the watermark (v,p,r)
   %embed in 2x2 subblock
   n=1;
   for ii=y1:y1+(blocksize/2)-2
      for jj=x1:x1+(blocksize/2)-2
         block(ii,jj)=bitset(block(ii,jj),1,embedbit(n));
         n=n+1;
      end
   end

  % block(x1:x1+(blocksize/2)-1, y1:y1+(blocksize/2)-1)=subblok;

   if (x1+(blocksize/2))> blocksize
      if y1 +(blocksize/2) < blocksize
         x1=1;
         y1= y1+(blocksize/2);
      end
      else
         x1=x1+(blocksize/2);

   end
end
 watermarked_image(y:y+blocksize-1, x:x+blocksize-1)=block;
  if (x+blocksize) > Nc
    if y+blocksize < Mc
       x=1;
       y=y+blocksize;
   end
    else
       x=x+blocksize;

   end
  end

difference= cover_object- watermarked_image;
imshow(difference,[-1 1])
%sub-block watermark generation
watermarked_image_int=uint8(watermarked_image);
imwrite(watermarked_image_int,'ustest.bmp','bmp');


% display processing time
%elapsed_time=cputime-start_time,
```

```
% display psnr of watermarked image
psnr=psnr(cover_object,watermarked_image),

% display watermarked image

imview(watermarked_image_int)
title('Watermarked Image')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name:                  Jasni Zain
%Project:       Level-2 detection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

% save start time
start_time=cputime;

blocksize=8;                % set the blocksize


% read in the cover object
file_name='tamper250.bmp';
cover_object=(imread(file_name));

%cover_object=rgb2gray(cover_object);
cover_object=double(cover_object);
% determine size of cover image
Mc=size(cover_object,1);            %Height
Nc=size(cover_object,2);            %Width
Br=floor(Nc/blocksize);  % Blocks per row
Bc= floor(Mc/blocksize); % Blocks per column
 ABB=[];
 numblock= Br*Bc; % number of blocks
 k=max(primes(numblock/2));
 for A= 1:numblock
    AB = mod((k*A), numblock)+1; %mapping the blocks
    ABB=[ABB AB];
 end
 B=[1:numblock];
 mapA= [B;ABB];


 mapB=[];
 for i= 1:numblock
    mapB(1,mapA(2,i))=mapA(2,i); %mapping the blocks
    mapB(2,mapA(2,i))=mapA(1,i);

 end
 mapB;

% determine maximum message size based on cover object, and blocksize
max_message=Mc*Nc/(blocksize^2);
```

```
    if bit==1
       par=par+1;
    end
end

if  rem (par,2)==0 %even
    p1= 1;
else p1=0;
end

if and(p1==bitp, v1==bitv)
   for ii= 1:(blocksize/2)
      for jj=1:(blocksize/2)
         subblok(ii,jj)=subblok(ii,jj);

      end
   end

   block(y1:y1+(blocksize/2)-1, x1:x1+(blocksize/2)-1)=subblok;
   if (x1+(blocksize/2))> blocksize
      if y1 +(blocksize/2) < blocksize
      x1=1;
      y1= y1+(blocksize/2);
      end
   else
      x1=x1+(blocksize/2);

    end
else
%find recovery block

x1=1;y1=1;
 for l=1:4

   substart=startblock(y1:y1+(blocksize/2)-1, x1:x1+(blocksize/2)-1);
   n=1;
   targetbit=0;
   for ii=y1:y1+(blocksize/2)-2
      for jj=x1:x1+(blocksize/2)-2

         bit=bitget(startblock(ii,jj),1);
         targetbit=bitset(targetbit,8-(n-3), bit);

        n=n+1;
      end
  end
```

```
    targetbit=bitset(targetbit,10, 0);
    targetbit=bitset(targetbit,9, 0);

      for ii= 1:(blocksize/2)
         for jj=1:(blocksize/2)
            subblok(ii,jj)=targetbit;
         end
      end
      block(y1:y1+(blocksize/2)-1, x1:x1+(blocksize/2)-1)=subblok;
       tamperblock=tamperblock+1;
      if (x1+(blocksize/2))> blocksize
         if y1 +(blocksize/2) < blocksize
         x1=1;
         y1= y1+(blocksize/2);
         end
      else
      x1=x1+(blocksize/2);

      end
   end

   end



end
   recover_image(y:y+blocksize-1, x:x+blocksize-1)=block;

      if (x+blocksize) > Nc
         if y+blocksize < Mc
            x=1;
            y=y+blocksize;
         end
      else
      x=x+blocksize;

      end
 end


%sub-block watermark generation
recover_image_int=uint8(recover_image);
imwrite(recover_image_int,'recovered250.bmp','bmp');

tamperblock
% display processing time
%elapsed_time=cputime-start_time,
```

```
% display psnr of watermarked image
%psnr=psnr(cover_object,watermarked_image),

% display watermarked image
figure
imshow(recover_image,[0 255])
title('Tamper detect')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name:                  Jasni Zain
%Project:       Image recovery
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

% save start time
start_time=cputime;

blocksize=8;                % set the blocksize


% read in the cover object
file_name='ustamp.bmp';
filename2='sig4.bmp';
sig_file=(imread(filename2));
sig_file=double(sig_file);
cover_object=(imread(file_name));

%cover_object=rgb2gray(cover_object);
cover_object=double(cover_object);
% determine size of cover image
Mc=size(cover_object,1);            %Height
Nc=size(cover_object,2);            %Width
Br=floor(Nc/blocksize);  % Blocks per row
Bc= floor(Mc/blocksize); % Blocks per column

ABB=[];
numblock= Br*Bc; % number of blocks
k=max(primes(numblock/2));
for A= 1:numblock
   AB = mod((k*A), numblock)+1; %mapping the blocks
   ABB=[ABB AB];
end
B=[1:numblock];
mapA= [B;ABB];


mapB=[];
for i= 1:numblock
   mapB(1,mapA(2,i))=mapA(2,i); %mapping the blocks
   mapB(2,mapA(2,i))=mapA(1,i);

end
mapB;
```

```
% determine maximum message size based on cover object, and blocksize
max_message=Mc*Nc/(blocksize^2);

% generate shell of watermarked image
recover_image=cover_object;
x=1;
y=1;

% process the image in blocks
%for block=1:(round(Mc/blocksize)* round(Nc/blocksize))

%sumblock=sum(sum( cover_object(x:x+blocksize-1,y:y+blocksize-1)));
%wm= round(sumblock/(blocksize*blocksize));
tamperblock=0;
for i = 1:numblock

   % numbering block
   block=cover_object(y:y+blocksize-1,x:x+blocksize-1);

   cover= RemoveLSB(block,1); % reset the LSB to 0
   AvgB=round(sum(sum(cover))/(blocksize*blocksize)); % average of block

   targetblock= mapA(2,i);
   rownum=round( ceil(targetblock/Br));
   colnum= round(targetblock-(rownum-1)*Br);
   ys=(rownum-1)*blocksize+1;
   xs=(colnum-1)*blocksize+1;
   startblock=cover_object(ys:ys+blocksize-1,xs:xs+blocksize-1);
   cover1=RemoveLSB(startblock,1);
   AvgBc=round(sum(sum(cover1))/(blocksize*blocksize));

   % prepare sub-block
   x1=1;
   y1=1;
   for sub= 1:4
      subblok= block(y1:y1+(blocksize/2)-1, x1:x1+(blocksize/2)-1);
      bitv= bitget(subblok(1,1),1);   % getting v from sub block
      bitp= bitget(subblok(1,2),1);   % getting p from sub block
      subblok=RemoveLSB(subblok,1);
      AvgBs=round(sum(sum(subblok))/(blocksize/2)*(blocksize/2));
      if     AvgBs >= AvgB
            v1=1;
      else    v1=0;
      end
      %parity
      par=0;
      embedbit=[];
```

```
for b=8:-1:2
   bit=bitget(AvgBs,b);
   embedbit=[embedbit bit];
   if  bit==1
      par=par+1;
   end
end

if   rem (par,2)==0 %even
    p1= 1;
else p1=0;
end

if and(p1==bitp, v1==bitv)
   for ii= 1:(blocksize/2)
      for jj=1:(blocksize/2)
         subblok(ii,jj)=subblok(ii,jj);

      end
   end
else
%find recovery block
  block=sig_file(y:y+blocksize-1,x:x+blocksize-1);
end

    %block(y1:y1+(blocksize/2)-1, x1:x1+(blocksize/2)-1)=subblok;

   if (x1+(blocksize/2))> blocksize
      if y1 +(blocksize/2) < blocksize
         x1=1;
         y1= y1+(blocksize/2);
      end
   else
      x1=x1+(blocksize/2);

   end
end
recover_image(y:y+blocksize-1, x:x+blocksize-1)=block;

   if (x+blocksize) > Nc
      if y+blocksize < Mc
         x=1;
         y=y+blocksize;
      end
   else
   x=x+blocksize;
```

```
    end
end


%sub-block watermark generation
recover_image_int=uint8(recover_image);
imwrite(recover_image_int,'us3.bmp','bmp');

tamperblock
% display processing time
%elapsed_time=cputime-start_time,

% display psnr of watermarked image
%psnr=psnr(cover_object,watermarked_image),

% display watermarked image
figure
imshow(recover_image,[0 255])
title('Tamper detect')
```

Appendix C – Recovered Images

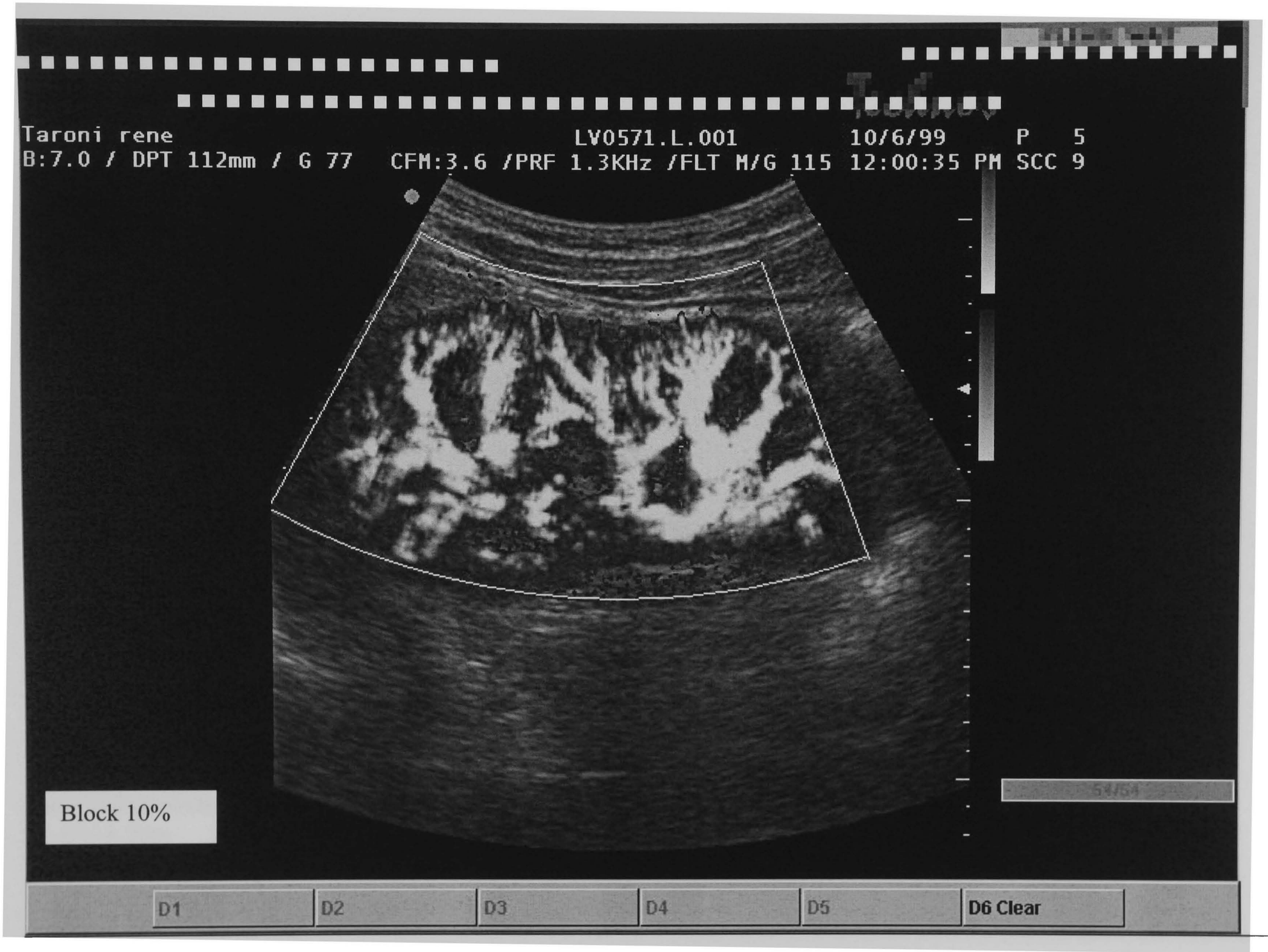

PLEASE WAIT
Taroni rene
LVO171.L.001
16/6/99
P 5
B:7.0 / DPT 112mm / G 77
CFM:3.6 PRF 1.3KHz /FLT M/G 115
11:00:35 PM
SCC 9
Spread 10%
D1
D2
D3
D4
D5
D6 Clear


PLEASE WAIT
Technos
LV0171.L.001
B:7.0 / DPT 112mm / G 77
CFM 3.6 PRF 1.3KHz /PLT M/G 15
Spread 20%
D1
D2
D3
D4
D5
Clear

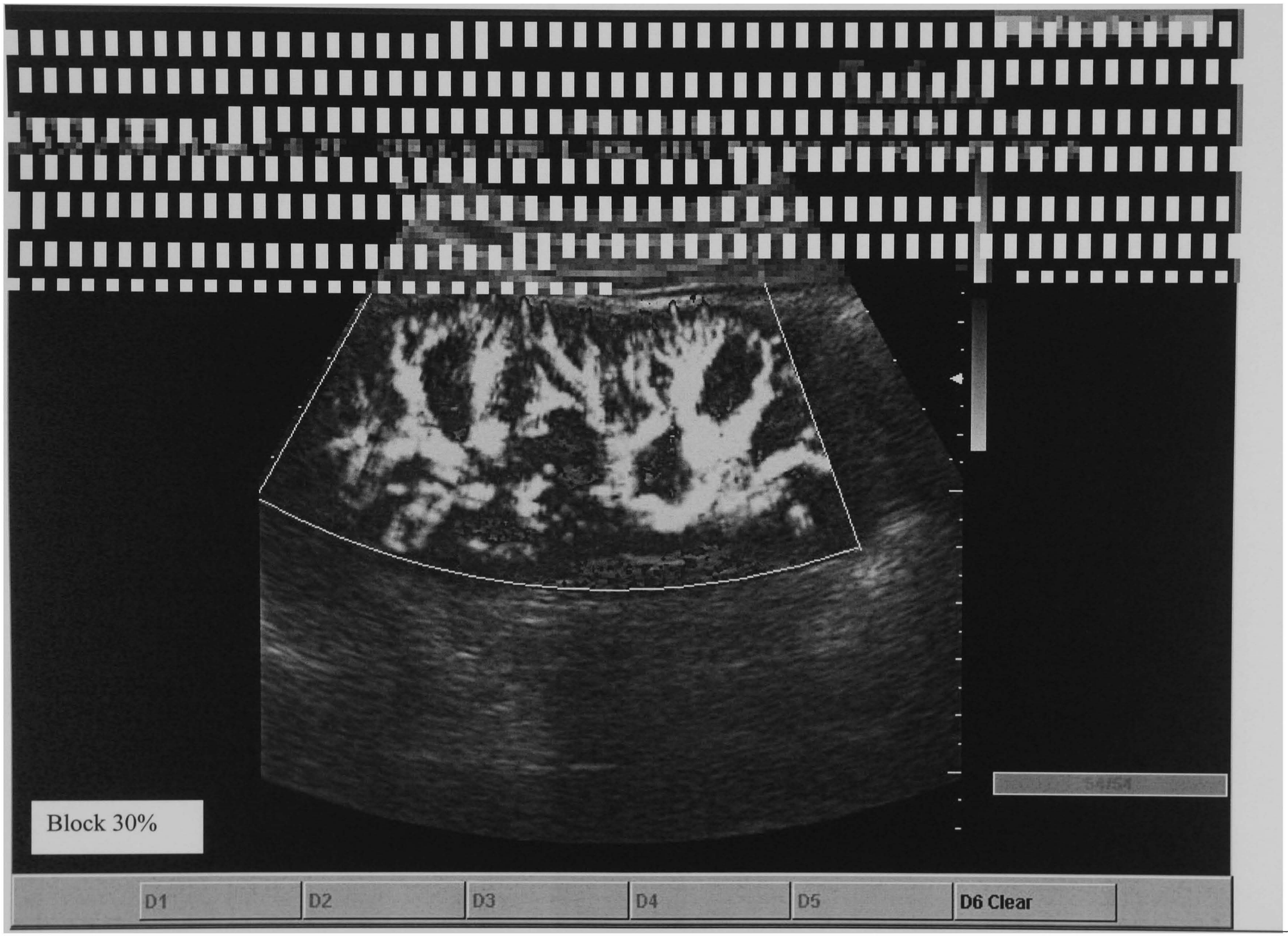

PLEASE WAIT
Technos
LV0571.L.001
10/6/99
P
5:7.0 / DPT 112mm / G 77
CFM:3.6 PRF 1.3KHz /FLT M/S 115 12:00:35 PM SCC 9
Spread 30%
D1
D2
D3
D4
D5
D6 Clear

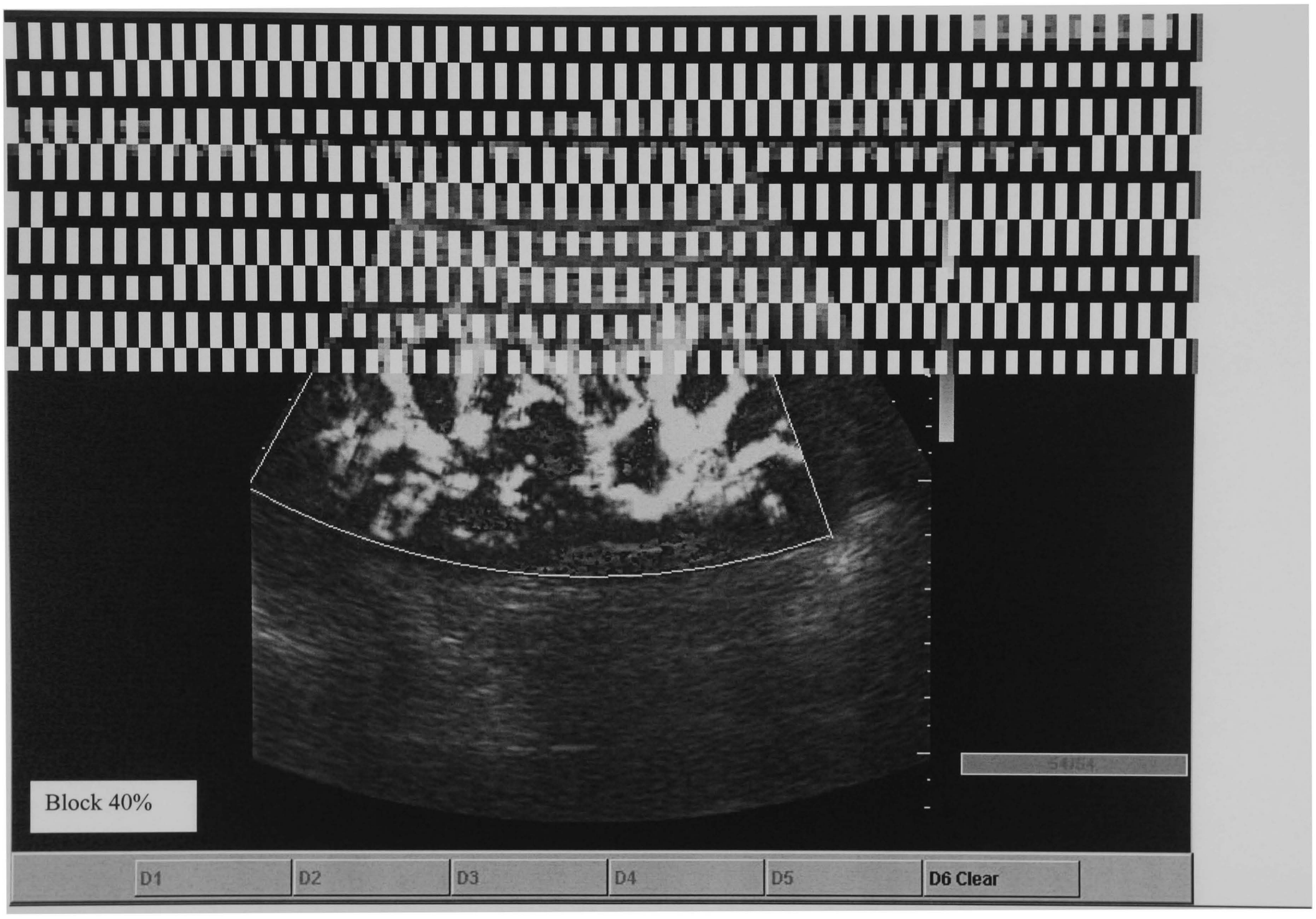

PLEASE WAIT
Technos
Spread 40%
D1
D2
D3
D4
D5
D6 Clear

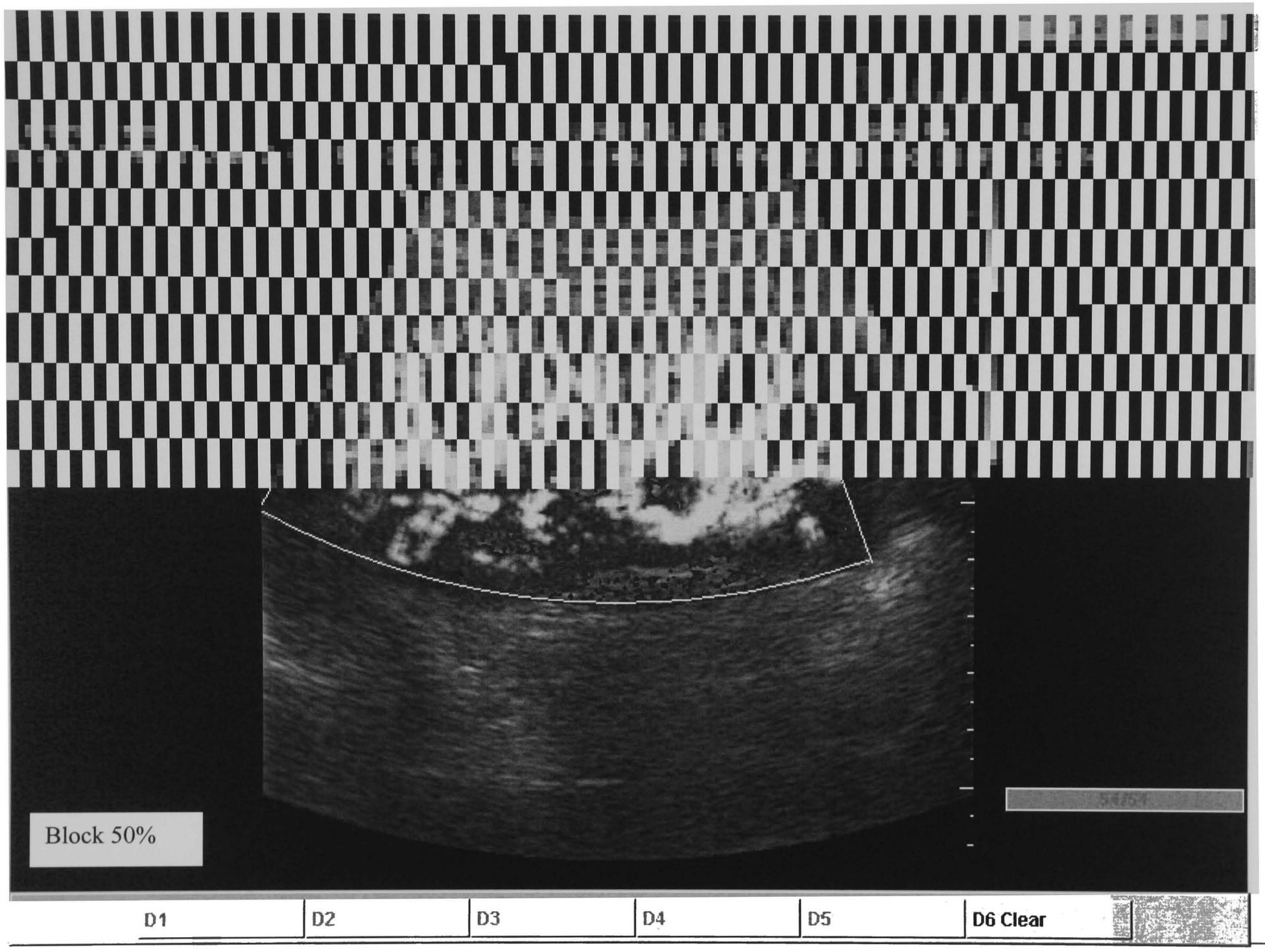

Spread 50%


Taroni rene
LV0571.L.001
10/6/99
P 5
B:7.0 / DPT 112mm / G 77
CFM:3.6 /PRF 1.3KHz /FLT M/G 115
12:00:35 PM SCC 9
Block 10%
D1
D2
D3
D4
D5
D6 Clear


Block 20%
D1
D2
D3
D4
D5
D6 Clear


Block 30%
D1
D2
D3
D4
D5
D6 Clear


Block 40%
D1
D2
D3
D4
D5
D6 Clear


Block 50%
D1
D2
D3
D4
D5
D6 Clear